**Лабораторная работа № 12**

**Тема:** Динамическое управление доступом на основе контекста

**Цель:** разработать систему, которая динамически адаптирует права пользователей в зависимости от контекста (времени доступа, IP-адреса).

**Время выполнения лабораторной работы (аудиторные часы):** 4 часа

**Оборудование и программное обеспечение:** ПК с установленной PostgreSQL, текстовый редактор для написания SQL-запросов.

**1. Теоретические сведения**

**1.1 Понятие динамического управления доступом**

Динамическое управление доступом (Dynamic Access Control) — это подход к управлению доступом, который учитывает контекстные факторы для принятия решений о предоставлении или ограничении доступа к данным или ресурсам. Это гибкая модель, которая позволяет повысить безопасность за счет анализа текущих условий, таких как: **время доступа**, **местоположение или IP-адрес**, **устройство или платформа пользователя**.

Основная цель динамического управления доступом — предотвращение несанкционированного доступа в реальном времени, основываясь на дополнительных параметрах, помимо роли пользователя.

Особенности динамического управления доступом:

1. Права доступа изменяются в зависимости от текущих условий.
2. Решения принимаются на основе реальной ситуации.
3. Учет дополнительных параметров снижает вероятность компрометации данных.

**1.2 Реализация динамического управления доступом в PostgreSQL**

Динамическое управление доступом в PostgreSQL позволяет адаптировать права пользователей в зависимости от текущих условий, таких как IP-адрес, время суток или роль пользователя. Рассмотрим основные подходы к реализации такого управления с использованием конфигурации доступа, функций и триггеров, а также политики безопасности на уровне строк (Row Level Security, RLS).

Одним из ключевых аспектов динамического управления доступом является ограничение доступа к базе данных на основании IP-адреса клиента. PostgreSQL предоставляет встроенный механизм управления доступом через файл конфигурации pg\_hba.conf (Host-Based Authentication). Этот файл определяет, каким пользователям и ролям разрешено подключение к базе данных, а также откуда (по IP-адресу или сети) они могут подключаться.

Настройка файла pg\_hba.conf

Файл pg\_hba.conf состоит из записей, определяющих правила подключения. Каждая запись включает следующие поля:

* Тип подключения (например, host для TCP/IP-соединений).
* Имя базы данных (или all, чтобы применить правило ко всем базам данных).
* Имя пользователя (или all, чтобы применить правило ко всем пользователям).
* IP-адрес клиента (или сеть в формате CIDR).
* Метод аутентификации (например, md5, password, reject).

Временные ограничения доступа особенно полезны для предотвращения несанкционированного доступа в нерабочее время или в определенные периоды. В PostgreSQL это можно реализовать с помощью функций и триггеров.

Функция проверяет текущее время перед выполнением операций. Если время выходит за рамки допустимого диапазона, операция прерывается с сообщением об ошибке.

CREATE OR REPLACE FUNCTION restrict\_by\_time()  
RETURNS TRIGGER AS $$  
BEGIN  
 IF EXTRACT(HOUR FROM CURRENT\_TIMESTAMP) NOT BETWEEN 9 AND 18 THEN  
 RAISE EXCEPTION 'Доступ разрешен только в рабочее время (9:00 - 18:00)';  
 END IF;  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;

Применение триггера

Триггер вызывает функцию перед выполнением операций INSERT, UPDATE или DELETE.

CREATE TRIGGER check\_access\_time  
BEFORE INSERT OR UPDATE OR DELETE ON employees  
FOR EACH ROW  
EXECUTE FUNCTION restrict\_by\_time();

PostgreSQL позволяет динамически ограничивать доступ к данным в зависимости от роли пользователя. Для этого можно использовать политики безопасности на уровне строк (Row Level Security, RLS).

RLS нужно включить для каждой таблицы, на которую распространяется политика:

ALTER TABLE employees ENABLE ROW LEVEL SECURITY;

Политика задает условия, при которых пользователи могут получить доступ к строкам таблицы. Например, следующая политика позволяет доступ к строкам только тем пользователям, которые принадлежат роли manager\_role:

CREATE POLICY manager\_access  
ON employees  
USING (current\_user IN (SELECT rolname FROM pg\_roles WHERE rolname = 'manager\_role'));

Чтобы разрешить доступ только пользователям, подключенным с определенного IP-адреса, можно использовать функцию inet\_client\_addr():

CREATE POLICY ip\_based\_access  
ON employees  
USING (inet\_client\_addr() = '192.168.1.100');

Если пользователь или его IP-адрес не соответствуют условиям политики, PostgreSQL вернет пустой результат или ошибку.

**2. Задание для самостоятельного выполнения работы**

1. Создайте пользователей и роли, которые будут использоваться для проверки доступа. Назначьте пользователей к соответствующим ролям.
2. Настройте файл pg\_hba.conf для ограничения доступа на основе IP-адреса.
3. Проверьте, что доступ с разрешенных IP-адресов возможен, а с других — запрещен.
4. Напишите функцию и триггер для ограничения операций в зависимости от времени суток.
5. Проверьте корректность работы триггера, выполняя запросы в разные временные интервалы.
6. Включите RLS для таблицы и создайте политику, которая ограничивает доступ к строкам на основе роли пользователя или IP-адреса.
7. Протестируйте политику, выполняя запросы от имени разных пользователей и ролей.
8. Выполните тестирование созданной системы, чтобы убедиться, что доступ предоставляется или ограничивается в зависимости от контекста (времени, IP-адреса, роли).
9. Проанализируйте, насколько разработанная система устойчива к ошибкам и атакам.

Таблица 1 – Варианты заданий для самостоятельного выполнения

|  |  |
| --- | --- |
| **Вариант** | **Задание** |
| **1.** | Создайте таблицу **clients** с полями client\_id, name, email. Настройте доступ только для IP-адресов из сети 192.168.1.0/24. Реализуйте триггер для ограничения операций добавления данных в нерабочее время (9:00–18:00). |
| **2.** | Создайте таблицу **orders** с полями order\_id, client\_id, order\_date. Настройте RLS-политику, разрешающую доступ только пользователям роли sales\_role. Ограничьте операции обновления данных только для рабочего времени (9:00–17:00). |
| **3.** | Создайте таблицу **products** с полями product\_id, product\_name, price. Настройте доступ для пользователей из IP-адресов сети 10.0.0.0/8. Реализуйте триггер, запрещающий удаление записей в выходные дни. |
| **4.** | Создайте таблицу **employees** с полями employee\_id, name, position. Настройте RLS-политику, чтобы строки были доступны только пользователям с соответствующей ролью (manager\_role). Ограничьте добавление записей только на определенные часы (10:00–16:00). |
| **5.** | Создайте таблицу **transactions** с полями transaction\_id, amount, transaction\_date. Настройте доступ для IP-адресов из сети 172.16.0.0/12. Реализуйте функцию, запрещающую изменения записей в нерабочее время и в праздничные дни. |
| **6.** | Создайте таблицу **inventory** с полями item\_id, product\_id, quantity. Настройте RLS-политику, разрешающую доступ только пользователям с ролью warehouse\_role. Реализуйте триггер для предотвращения изменений данных вне рабочего времени. |
| **7.** | Создайте таблицу **shipments** с полями shipment\_id, order\_id, shipment\_date. Ограничьте доступ к данным для пользователей из IP-адреса 192.168.10.1. Реализуйте RLS-политику, чтобы пользователи могли видеть только данные, относящиеся к их роли. |
| **8.** | Создайте таблицу **sales** с полями sale\_id, product\_id, quantity, sale\_date. Реализуйте RLS-политику для ограничения доступа только для пользователей роли sales\_role. Настройте триггер для запрета добавления данных в выходные дни. |
| **9.** | Создайте таблицу **projects** с полями project\_id, project\_name, budget. Настройте доступ только для пользователей с IP-адреса 10.1.1.1. Ограничьте операции добавления и изменения данных только на рабочее время (9:00–18:00). |
| **10.** | Создайте таблицу **invoices** с полями invoice\_id, client\_id, total\_amount. Настройте RLS-политику, чтобы доступ был разрешен только пользователям роли billing\_role. Реализуйте функцию для ограничения операций после 18:00. |

**Контрольные вопросы**

1. Что такое динамическое управление доступом, и в каких ситуациях оно используется?
2. Как настроить доступ к базе данных PostgreSQL на основе IP-адреса?
3. Какие методы позволяют ограничить доступ к таблицам PostgreSQL по времени?
4. Что такое Row Level Security (RLS), и как его использовать для ограничения доступа?
5. Как с помощью триггеров можно реализовать временные ограничения операций в PostgreSQL?
6. Какие функции PostgreSQL позволяют определять текущий IP-адрес и время запроса?
7. Чем отличается динамическое управление доступом от статических моделей контроля доступа?
8. Какие потенциальные угрозы могут быть связаны с динамическим управлением доступом, и как их минимизировать?
9. Как проверить корректность настроек RLS для таблицы?
10. Почему комбинирование методов управления доступом (по IP, времени и роли) повышает уровень безопасности базы данных?

**Литература**

1. Blog PostgreSQL. Using sepgsql for Mandatory Access Control. URL: https://www.postgresql.org/
2. Агафонов В.В. Безопасность баз данных: учебное пособие. — М.: Юрайт, 2021. — 340 с.
3. Липаев В.В. Основы информационной безопасности. — М.: Академия, 2019. — 432 с.
4. PostgreSQL Security: Role and Privilege Management. URL: https://severalnines.com/blog/postgresql-security
5. Using GRANT and REVOKE in PostgreSQL. Digital Ocean Community. URL: https://www.digitalocean.com/community/tutorials/